ARMY RESEARCH LABORATORY

Network Device Checklist Automator

by Donald A. Bennett and Aaron P. Hiltgen

ARL-TR-5394 **November 2010**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TR-5394 **November 2010**

Network Device Checklist Automator

Donald A. Bennett and Aaron P. Hiltgen
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2010	Final	May 24 to August 2010

4. TITLE AND SUBTITLE
Network Device Checklist Automator

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Donald A. Bennett and Aaron P. Hiltgen

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CIN-S
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-5394

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Department of Defense (DoD) 8500 series mandates that all agencies implement the Security Technical Implementation Guides (STIGs) released by the Defense Information Systems Agency (DISA) to protect information systems against attackers and misuse. Agencies are required to perform regular checks on all their systems for compliance with these regulations. To make the STIG compliance validation process easier, the DoD allows the use of Security Readiness Review Scripts (SRRS), which automatically perform many checks, allowing auditors to focus attention on critical areas. Per the DoD, network devices are the most critical, but to date there are no commonly available Government automation tools for network devices such as routers, firewalls, switches, and intrusion detection systems (IDSs). Without sufficient support, many of these devices are running with little or no checking. The Network Device Checklist Automator (NDCA) seeks to become the first SRRS for network devices and provide the groundwork for future development. Our goal is to create a framework and implement full support for a few devices to demonstrate proof of concept in hopes of transitioning the project to other organizations for further evaluation and testing, and eventual implementation by all sections of the DoD tasked with ensuring network device STIG compliance.

15. SUBJECT TERMS
Security Technical Information Guide, Security Readiness Review Scripts

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Donald A. Bennett
Unclassified	Unclassified	Unclassified	UU	32	19b. TELEPHONE NUMBER (*Include area code*) (301) 394-5496

Contents

List of Figures

1. Introduction

Security Technical Implementation Guides (STIGs) released by the Defense Information Systems Agency (DISA) are used as a standard guideline for the configuration of information systems within the Department of Defense (DoD). A specific device can be tested for STIG compliance either manually by downloading the device's configuration information and manually checking rules in the relevant STIG against the device or automatically by using Security Readiness Review Scripts (SRRS). SRRS are tools that have been specifically designed to automate as much of the STIG compliance validation process as possible, significantly speeding up the entire process as a result. According to DISA's Information Assurance Support Environment (IASE), SRRS are available for "all operating systems and databases that have STIGs, and Web servers using IIS (*1*)." SRRS cover the STIGs for a wide variety of devices, excluding one major device type, network devices. Commercial off-the-shelf (COTS) solutions for network device validation, while available, are difficult to find and are very expensive. There is currently no Government-off-the-shelf (GOTS) SRRS identified or made available by DISA for network devices.

To remedy this situation, a project was launched with the goal of creating a program as proof of concept that a GOTS SRRS for network devices could be successfully implemented. The Network Device Checklist Automator (NDCA) is the result of that project. The program is able to apply rules created from a STIG for a specific network device to one or many configuration files for devices of that type. The program generates a report that includes all the necessary information on each rule in the STIG, allowing for an agent to quickly make a decision on compliance. Applicable sections of the configuration file are provided as output in the report for each rule, if such sections exist, and the program determines whether or not a rule is violated, if such a determination can be made by the NDCA.

STIGs, upon which the NDCA program is based, are discussed at the outset of the report. Next, the two major portions of the NDCA program are covered. First is the NDCA program that includes a graphical user interface (GUI) for agent interaction and the second section covers the master rule set and checklists, which are directly related to DISA's STIGs. When using the NDCA program, the agent chooses the master rule set and the specific checklist to perform its STIG compliance validation and generate reports. The rules are created based on information provided in the STIGs, with additional components based directly on regular expressions (Regex). The report goes on to outline the compliance validation report file that is created by the NDCA program, and ends with a discussion of ways that the program can be enhanced in the future.

2. Security Technical Implementation Guide (STIG)

2.1 STIGs

STIGs, created by DISA, are used as guidelines for secure configuration of Government information systems. Each STIG begins with a title page, as shown in figure 1, which includes the device covered by the STIG, the version number, the release date, and other pertinent information. Following the title page, the STIG lists each rule that must be adhered to by the targeted device to ensure network security. Figure 2 shows a sample rule from a STIG. Each STIG follows the same format, and the rules defined in each share a universal numbering system. This means that if the same rule is applicable to different versions of a device or multiple devices, portions of the rule (including the Group ID, Group Title, and Vulnerability Discussion fields) will be the same for all STIGs in which the rule appears. The Group ID, Group Title, Severity, Vulnerability Discussion, and Check Content fields from each STIG rule are used to create rules in the master rule set.

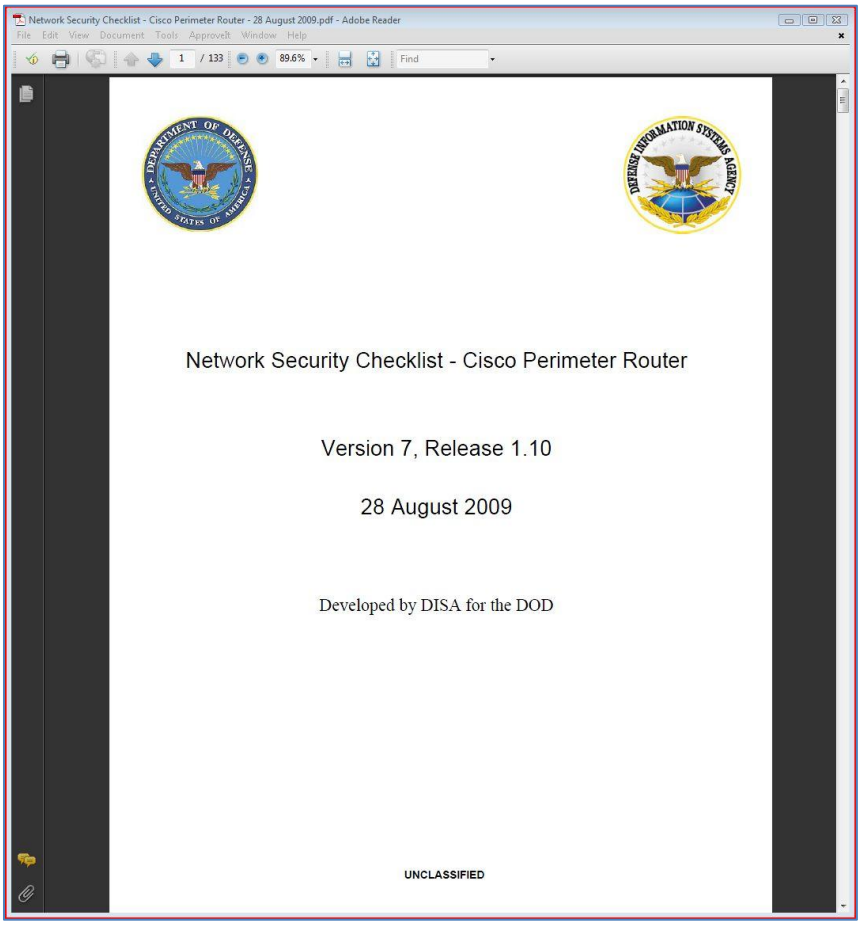


Figure 1. Sample STIG title page.

Group ID (Vulid): V-4622
Group Title: AG ingress ACL is not configured to secure enclave
Rule ID: SV-4622r5_rule
Severity: CAT I
Rule Version (STIG-ID): NET0162
Rule Title: The IAO/NSO will ensure premise router interfaces that connect to an AG (i.e., ISP) are configured with an ingress ACL that only permits packets with destination addresses within the site's address space.

Vulnerability Discussion: Any enclave with one or more AG connections will have to take additional steps to ensure that neither their network nor the NIPRNet is compromised. Without verifying the destination address of traffic coming from the site's AG, the premise router could be routing transit data from the Internet into the NIPRNet. This could also make the premise router vulnerable to a DoS attack as well as provide a backdoor into the NIPRNet. The DOD enclave must ensure that the premise router's ingress packet filter for any interface connected to an AG is configured to only permit packets with a destination address belonging to the DOD enclave's address block.

Responsibility: Information Assurance Officer
IAControls: ECSC-1

Check Content:
Review the running config of the router that connects to an AG and verify that each permit statement of the ingress ACL is configured to only permit packets with destination addresses of the site's NIPRNet address space or that belonging to the address block assigned by the AG network service provider.

Fix Text: Insure the ingress ACL for any interface connected to an AAG is configured to only permit packets with a destination address belonging to the sites address block.

Figure 2. Sample STIG rule.

3. GUI

A simple and intuitive GUI is essential to a universally accessible program (figure 3). We made the main process as autonomous as possible; it only asks the user for a few key pieces of data before generating the report.

Figure 3. Splash screen from program start.

If the master rule set is not in the same directory as the executable, the program will require the user to locate it before continuing (figure 4).

Figure 4. Warning that a master rule set is required to use the program.

The first time the NDCA is run, the user must provide a NDCA checklist and the directory containing the configuration files (figure 5). This can be accomplished via the Sources tab at the top.

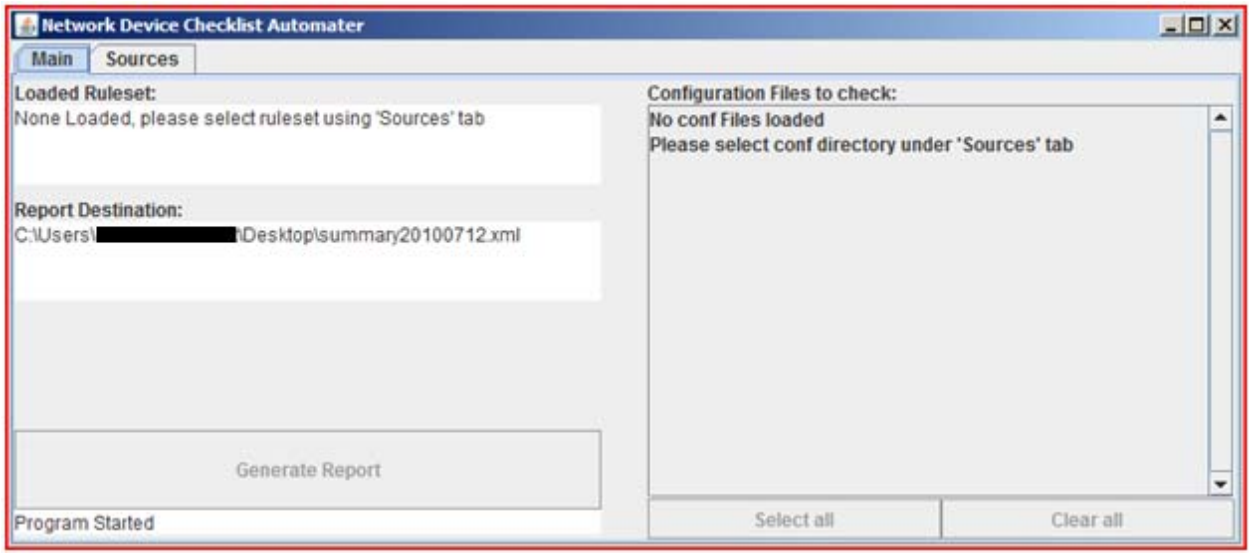


Figure 5. The main screen on a fresh load without any options selected.

The user needs to select the destination of the report file, the checklist of rules specific for the device the user wants to audit, and a directory where one or more configuration files is located (figure 6). There is also support for filtering files so that only dumps appear.

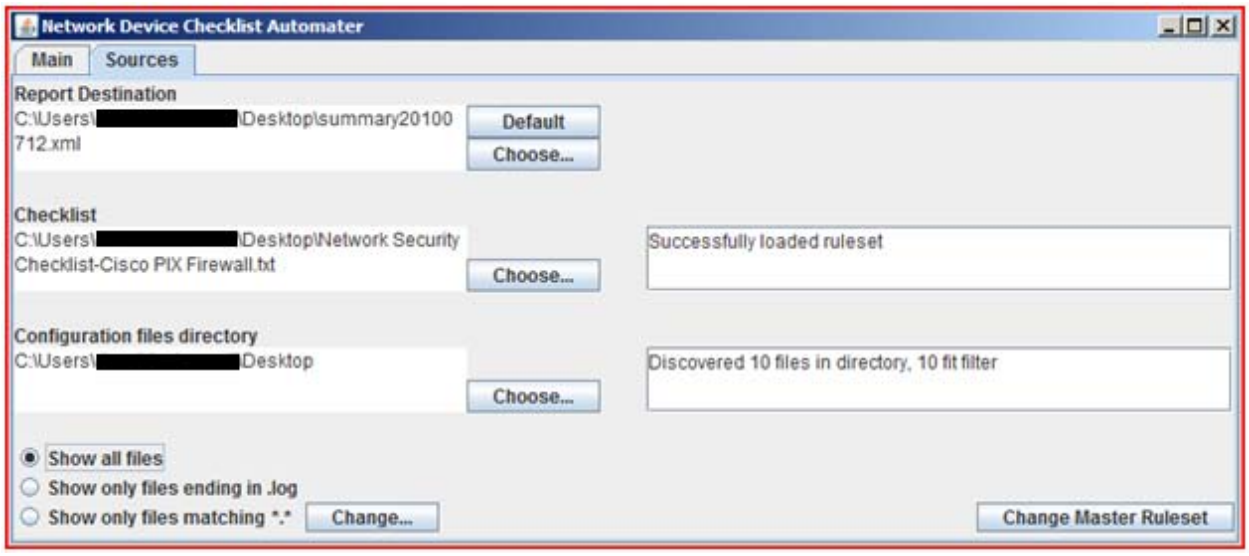


Figure 6. Sources tab with options selected.

The built-in file browser enables easy file and folder selection (figure 7).

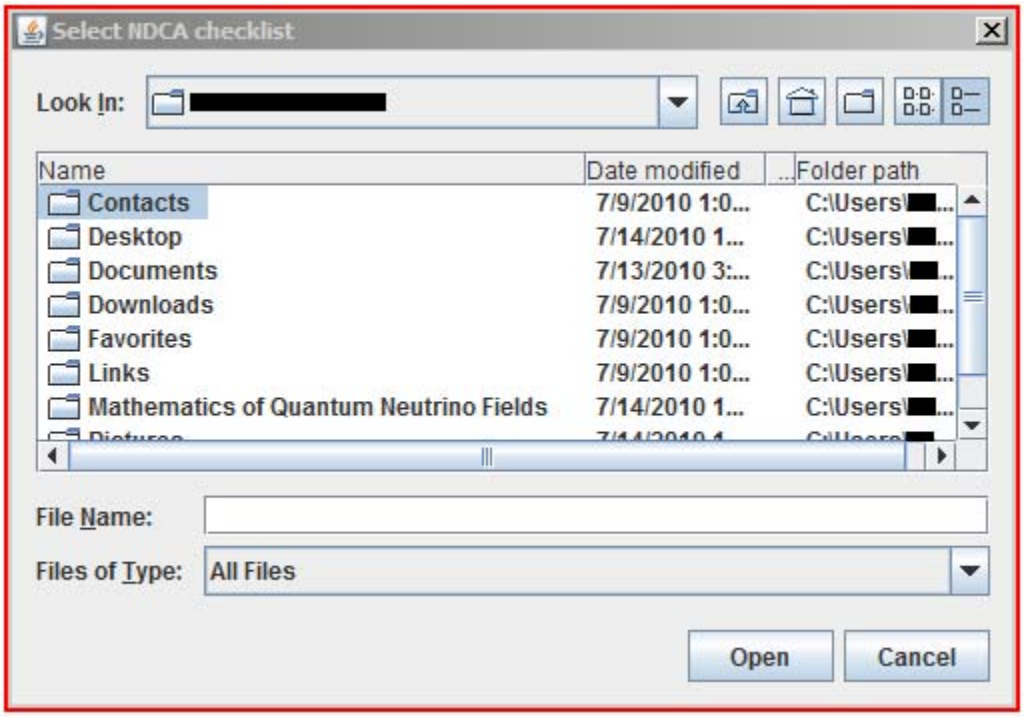


Figure 7. Generic file browser for choosing options.

At that point, users can select which configuration files they wish to include via checkboxes, and then click the button to generate the report (figure 8).

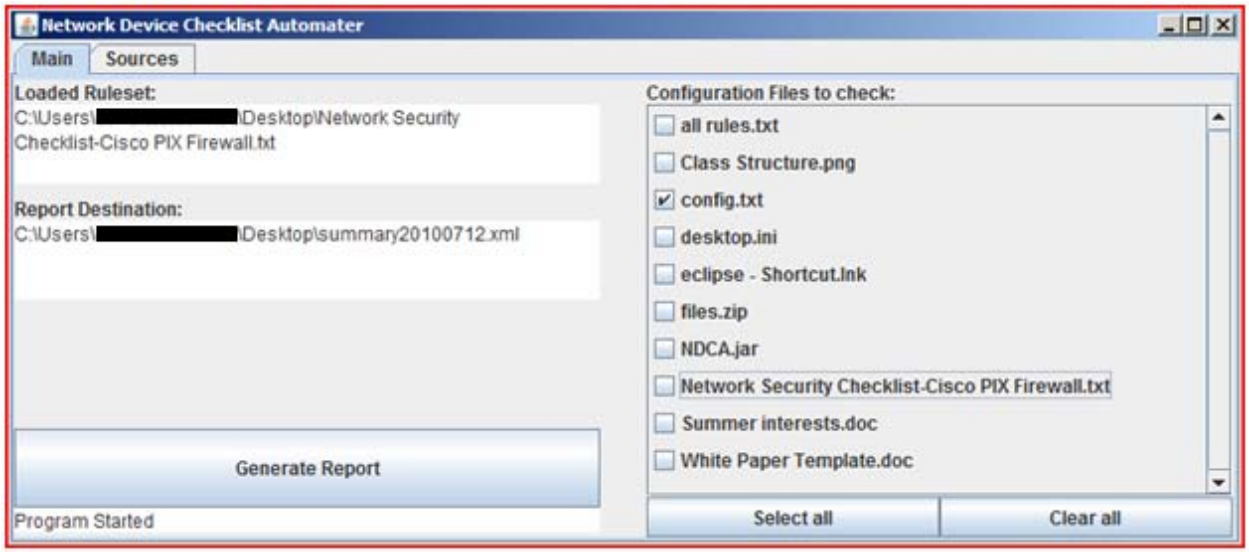


Figure 8. Main screen with options selected, ready to run.

The NDCA pulls in the configuration files selected and runs them all against the rule set in batches before pushing the results out to a report (figures 9 and 10).

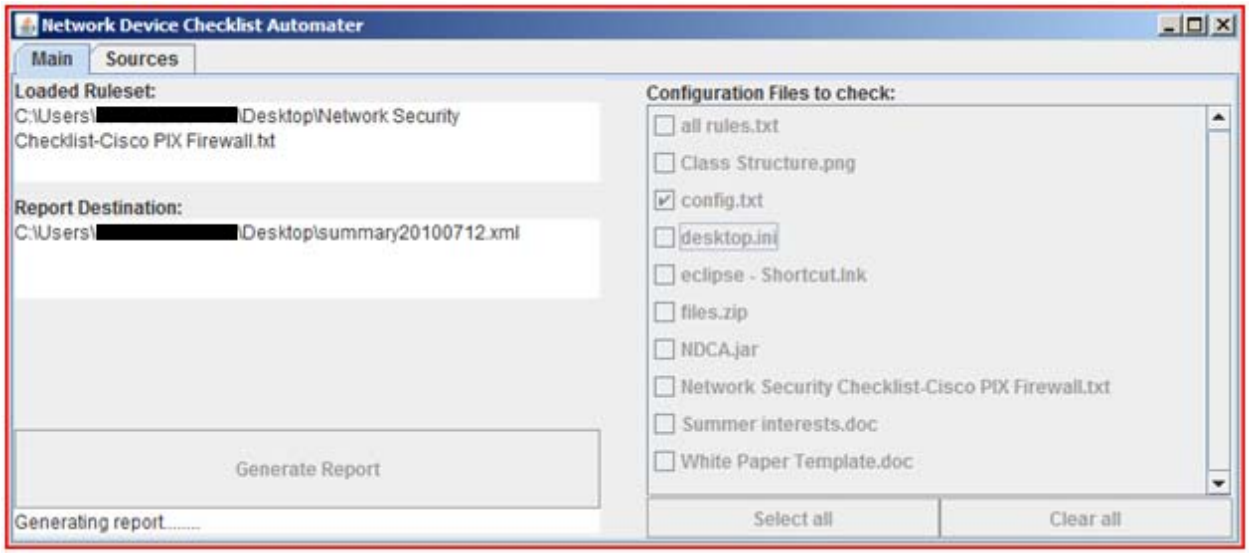


Figure 9. Main screen while generating a report.

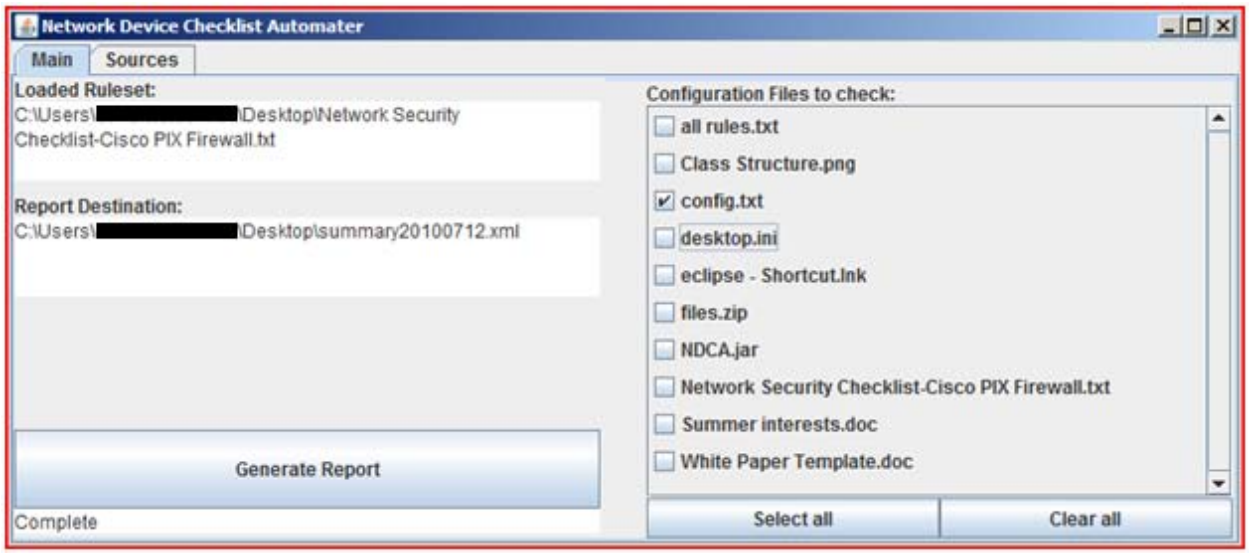


Figure 10. Main Screen after completion of the report.

One important distinction between the master rule set and the checklists is that an inability to completely parse the master rule set results in failure, while checklists can recover from small errors by ignoring lines. We chose to put tighter restrictions on the master rule set because any errors would propagate to all reports and checklists, while a single checklist can only influence a single report. It is important to note that all errors are reported, so the user is always aware if there is an issue, even if it does not result in a failure.

The report includes statistics about individual configurations and about the group as a whole. By default, the report is stored in the same location as the executable, under the name "summary<date>.xml." The date is an eight-digit number corresponding to the date on the host computer, in the format "ddmmyyyy." The NDCA also has a strict protocol that prevents it from overwriting or editing files that it did not create; consequently, it will add a number to the end of the filename if the target file already exists. This feature protects reports and configuration files from accidental loss or damage.

The NDCA currently only supports loading one device's checklist file at a time, as there is no simple way to determine which configuration files are from which devices. It would not be difficult to add support for multiple checklist files at once, but it would require that something be added to the configuration files to denote the device from which the files have been dumped. At his point, we decided that it was important for the NDCA to be compatible with current collection methods since it is still a supplement to the auditing process.

Whenever the program is closed, it takes a few seconds to store its state and user selections to a file, which is loaded on subsequent runs. This is done through a process called serialization, which converts objects to and from data streams. This means that it may only be necessary to make changes to the default settings once.

4. Rule Sets

The initial design of the rule set was to give each device a file that would contain all the information required to do the check. In this way, they would be self-contained and modular. However, when creating these files, we noticed that many of the rule sets have a high degree of overlap between devices, with little if any change between instances. Because of this discovery, we decided to change to a database model. Instead of making a complete file for each device, the bulk of the information is now stored in a universal database file, referred to as the *master rule set*, and the individual device files, known as *checklists*, contain only references to this repository. The main benefit to this is that we only need to write a rule once and can then reference it in as many rule sets as we want, saving time in rule set development and storage space by eliminating redundancy. The decision to shift to a central rule repository also has the side-effect of propagating any changes made to an entry instantly to all affected rule sets. This allows us to make updates or improvements once and use them many times. This paradigm also reduces system maintenance and increases the accuracy by ensuring that all checks are done using identical rule implementations. Lastly, the user experience changed so that the initial load on program start is long (when the master rule set is loaded), but individual device rule sets load almost instantaneously (since they contain only pointers). The first implementation had a quick program launch, but changing between device rule sets caused the program to hang noticeably as it reparsed each rule.

4.1 Master Rule Set

Currently, the master rule set is a plain-text file that stores a complete version of all the rules included in the various STIGs for network devices. It acts as a central database of rules, allowing changes to a rule to propagate easily across all related STIGs. The master rule set is split into two sections, flags and rules. Flags are used by the NDCA program to distinguish between multiple versions of a rule. If two or more devices share a rule, a distinct version of the rule must be created for each instance based on the device's unique configuration file. By setting a flag for the rule, the NDCA can distinguish which version of the rule to use based on the checklist that is chosen by the operator.

At the beginning of the master rule set file is the header #Flags, as shown in figure 11. Underneath this header is the flag hierarchy. Each flag is related to its "parent" flag in the form child→parent, which allows the NDCA program to locate a version of the relevant rule in all instances. If the version of a rule referenced by a flag does not exist, the program looks for a version of the rule referenced by the flag's parent. If that version does not exist, the process repeats for the parent's parent and so on until a valid version of the rule is found. Flags that do not have a parent defined in the flags field automatically have the default, non-flagged version of

the rule as their parent. In this manner, the program ensures that all rules are checked against the most relevant version of each rule that is available.

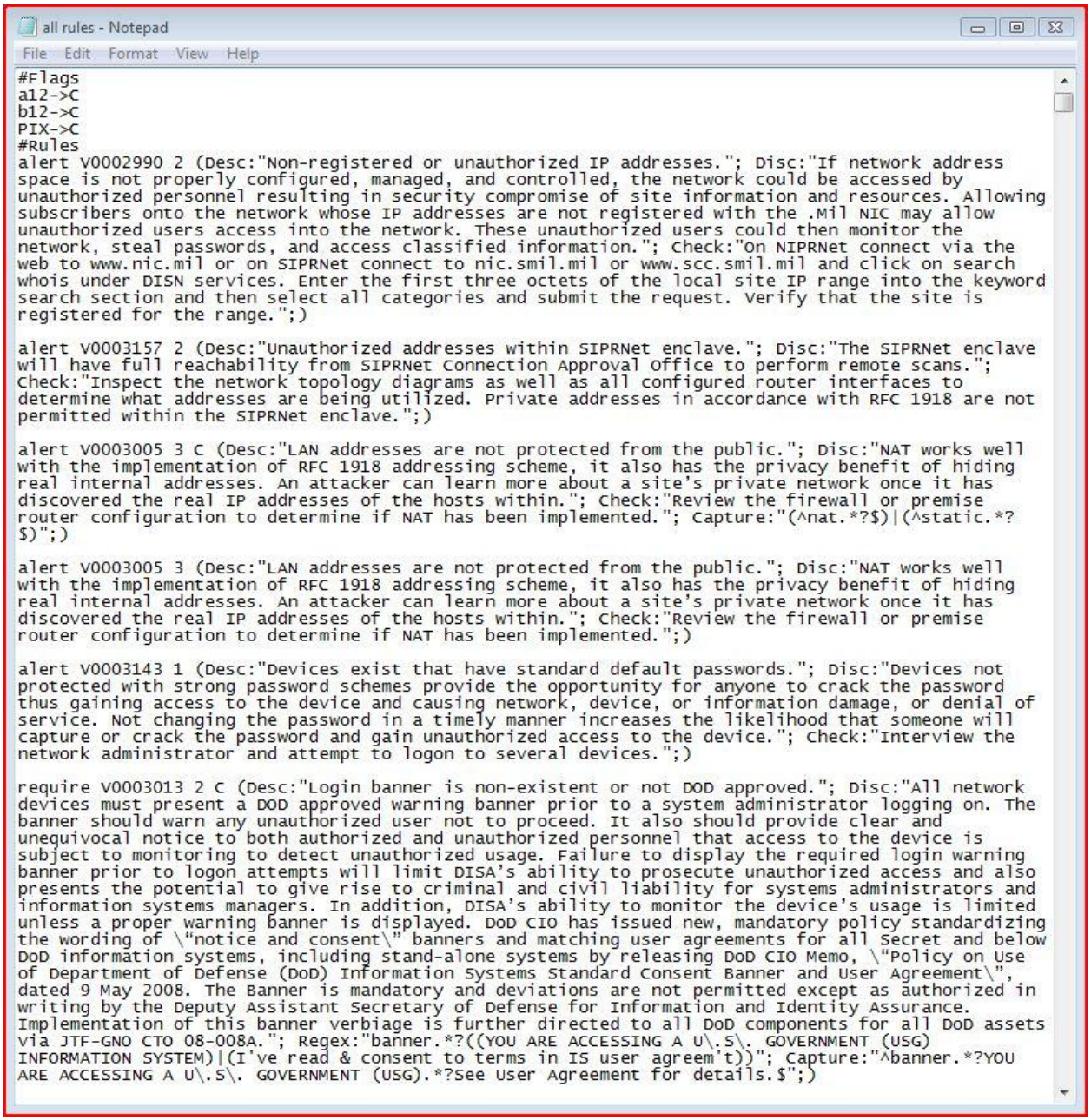


Figure 11. Example of a master rule set.

The second section of the master rule set begins with the #Rules header. This section of the file defines each rule in a specific format (as shown in figure 11). Each rule, or each version of a rule, begins with several fields that are required for every rule every time. The first field is one of two words, alert or require. Alert indicates that the NDCA should consider a rule violated if a specific line (or lines) of text is found in the configuration file. Require indicates that the NDCA

will consider a rule violated if a specific line (or lines) of text does not appear in the configuration file, i.e., the specified line of text is required to appear in the configuration file. Rules that cannot be determined to pass/fail based on text in the configuration file, such as those rules requiring the agent to interview the system administrator (SA), still must have either alert or require at the beginning of the rule, but it does not matter which one is chosen.

Following alert/require is the rule number, identified as the Group ID in the STIG. This is followed by a single digit, 1, 2, or 3, to indicate whether the rule is of category (CAT) I, II, or III severity level. The final required field is the flag field, used to indicate the flag, or device version, to which the rule is intended to apply. Note: The flag field is left blank for the default version of any rule. The remaining fields are optional based on what information is necessary for a particular rule or rule version. A completed rule should have the following format:

<alert|require> <Group ID> <Severity> [<flag>] ([Desc:“<field text>”;] [Disc:“<field text>”;] [Check:“<field text>”;] [Regex:“<field text>”;] [Capture:“<field text>”;])

The optional fields portion of a rule is enclosed in parentheses (), and each optional field begins with the field name followed by a colon followed by the field text enclosed in quotation marks and completed by a semicolon, e.g., Desc:“field text”;. The fields that can be included in the optional portion of a rule are the following:

- *Desc*: This field is always included and corresponds to the Group Title in the STIG.
- *Disc*: This field is always included and corresponds to the Vulnerability Discussion in the STIG.
- *Check*: This field is optional and is used for text that is included in the Check Content portion of the STIG.
- *Regex*: This field is optional and provides the NDCA with a Regex pattern used to determine whether a rule passes or fails.
- *Capture*: This field is optional and provides the NDCA with a second Regex pattern used to determine which lines of the configuration file are applicable to the rule.

The file is ended with the flag #End, once all rules have been entered.

Since it is possible for the same rule to be applicable across multiple STIGs, there is only one instance of the master rule set, which contains all possible instances of any rule. This allows updates to rules to be made once and propagate instantly across multiple devices.

4.2 Checklists

Though all rules are kept in a central repository, the master rule set, it is still necessary to determine which subset of these rules applies to a particular device. This is accomplished by using checklists. Each checklist is a plain-text file, following the same format as the master rule

set, that lists information pertinent to a specific device. A sample checklist is shown in figure 12. The first field on every checklist is #Device. Underneath that line is the name of the device, based on the device name in the title of the STIG. The next line is #Flag, which differs slightly from the #Flags field in the master rule set. In the checklist, only one flag is listed, which indicates the first version of each rule that the NDCA should try to find. This flag is applied to every rule, after which the NDCA looks for "parent" versions of the rule until a valid rule version is found. The final section of each checklist is the #Rules section, which is a simple list of every rule number (Group ID) found in the STIG. It is important that the rule numbers in the checklist exactly match the rule numbers in the master rule set. In fact, the NDCA notifies the user if a particular rule number is not found in any form. Once all the rules are listed, each checklist must end with a #End line.

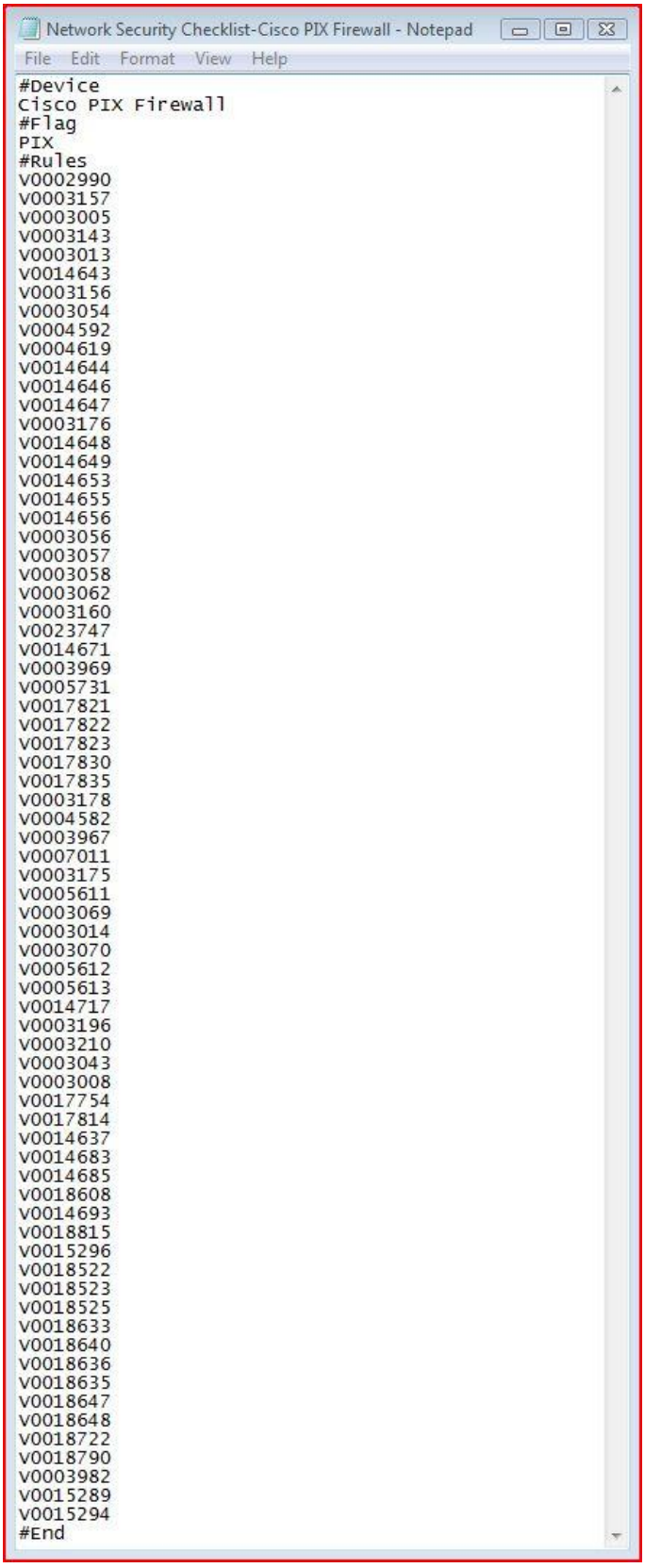


Figure 12. Example of a checklist.

4.3 Regular Expressions

The format of the rules themselves is inspired by Snort, a popular intrusion detection system (IDS) program. Snort uses Regex to check network traffic for patterns that might denote a problem, and we use the same idea to find configuration settings that might cause a security hole. Each rule is designed to contain all the data auditors would need to perform an evaluation on the system, making NDCA reports the primary auditing reference. Rules also contain an optional field that displays data directly from the configuration files in the report alongside their rules to again speed up manual checking.

The most important part of the NDCA is its use of Regex to match portions of a configuration file. Regex is how the NDCA determines pass/fail for a rule, and how it determines which portion(s) of a configuration file is related to each rule. Regex has specific rules that are used to create a "pattern" that can be used to find a substring within a block of text. While some advanced Regex concepts vary slightly between programming languages, all use Regex in some form for pattern matching. The NDCA and the master rule set use Perl-style Regex to create matching patterns. Basic Regex patterns allow for the use of wildcards for one or multiple consecutive characters, exclusive-or logic, and specifying a range of characters to check for. For example, instead of writing several lines of code to locate text, a user can write one Regex string, \b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b, which would locate any e-mail addresses in the targeted text. With slight modifications, the same Regex pattern can be used by a programmer to check that a user's input is a valid e-mail address. Regex is used in many applications, including text editors, command line text search functions (grep, egrep,…), programming compilers, and find/replace functions to speed the location of a specific pattern of text. Two Web sites, http://java.sun.com and http://www.regular-expressions.info, were invaluable sources of information on Regex throughout this project.

In the Regex field of a rule, a pattern is created for a specific line in the configuration file (e.g., ip cef). If that pattern appears in the configuration file, the NDCA is able to identify that line and take action based on whether the rule is an alert or require rule. The NDCA is also able to take action based on the rule if the line does not appear in the configuration file. A Regex can be difficult to craft since a small mistake in constructing a pattern can cause extremely unexpected results, and patterns often require creative methods to achieve the desired result. As a result, pattern creation for the master rule set should only be attempted by someone who is thoroughly trained in Regex in cooperation with a subject matter expert (SME) for the device covered by the STIG in question.

The NDCA currently only supports a single pass system for reasons of simplicity and speed, which has worked out well for most rules. It is also important to note that this system is not perfect; some checks are not easily adapted to Regex, while others require conditional statements and multiple passes through the configuration file. Due to time constraints, we decided not to

implement these features, but have tried to make it as simple as possible to add support at a later time.

5. Programming

The program is structured with modularity and extensibility in mind. As illustrated in figure 13, the GUI front end collects the data from the user and does some preliminary checks for validity. When the user has met all the requirements for generating a report, the GUI packages up the data and sends it to a report generator, which runs in its own thread. The report generator uses the architecture it has been given to pull in the rules as a RuleSet object. Finally, after parsing all the data, the report generator packages everything into an XMLWriter object, which generates the XML file and terminates the thread. The reason that we divided the program in this way was to make it as seamless as possible for later developers to change a module or put new plug-ins between existing nodes. The use of single information bundles as opposed to data streams also helps strengthen modularity and ensure a healthy dependence tree.

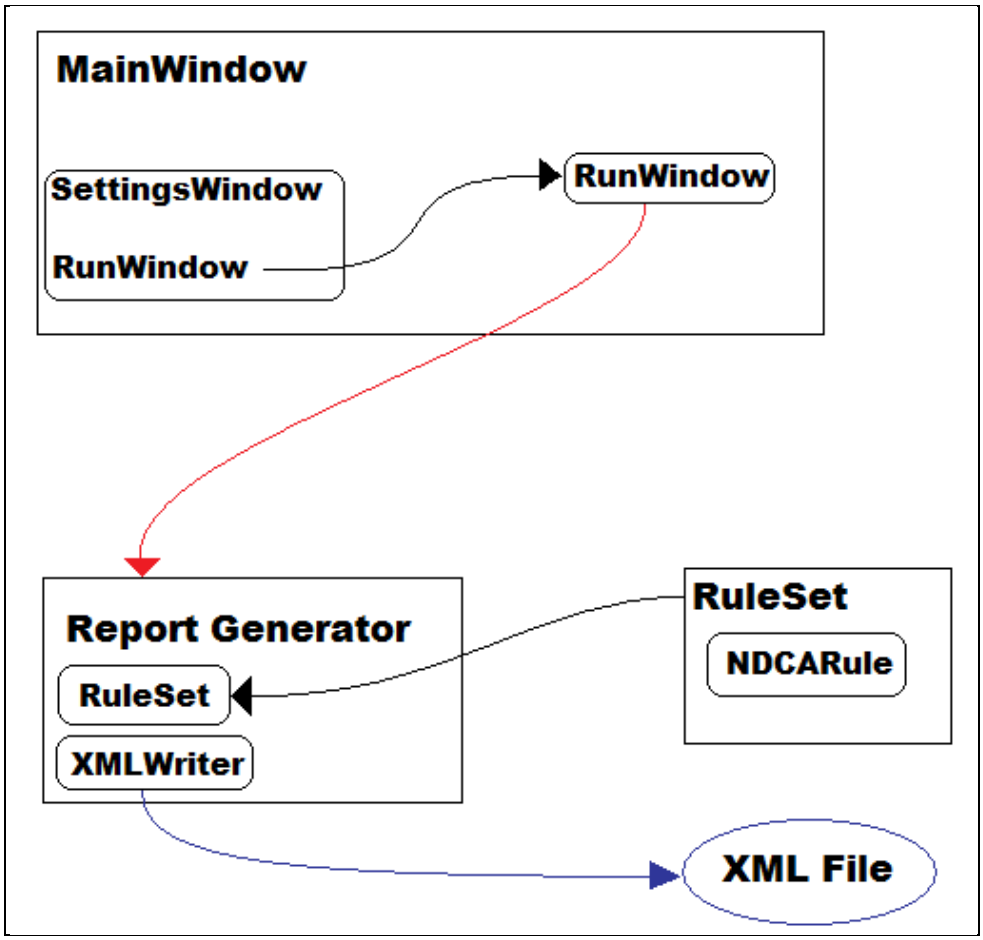


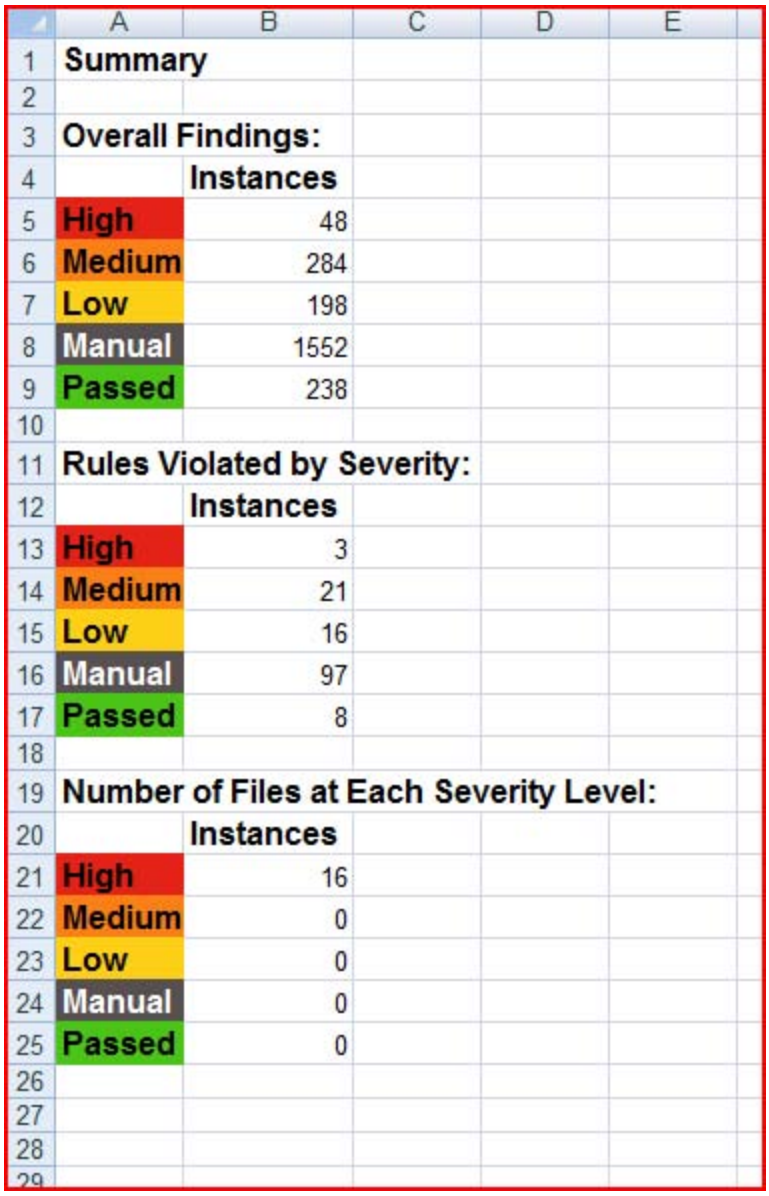
Figure 13. General control flow of program modules and data.

We decided that if the GUI crashed or was closed while the report generator was running, it would continue until it had completed its task. Since the program institutes a strict no-overwrite policy, it was considered unlikely that allowing the program to continue would cause any damage, but rather would make it more robust.

6. XML Report

Based on the success of similar projects, we decided to output findings in XML format so they could be displayed in a spreadsheet. The XML reports are generated manually and are very basic. We used other reports as a template to create our own spreadsheet that would display everything in a well-organized manner for the user.

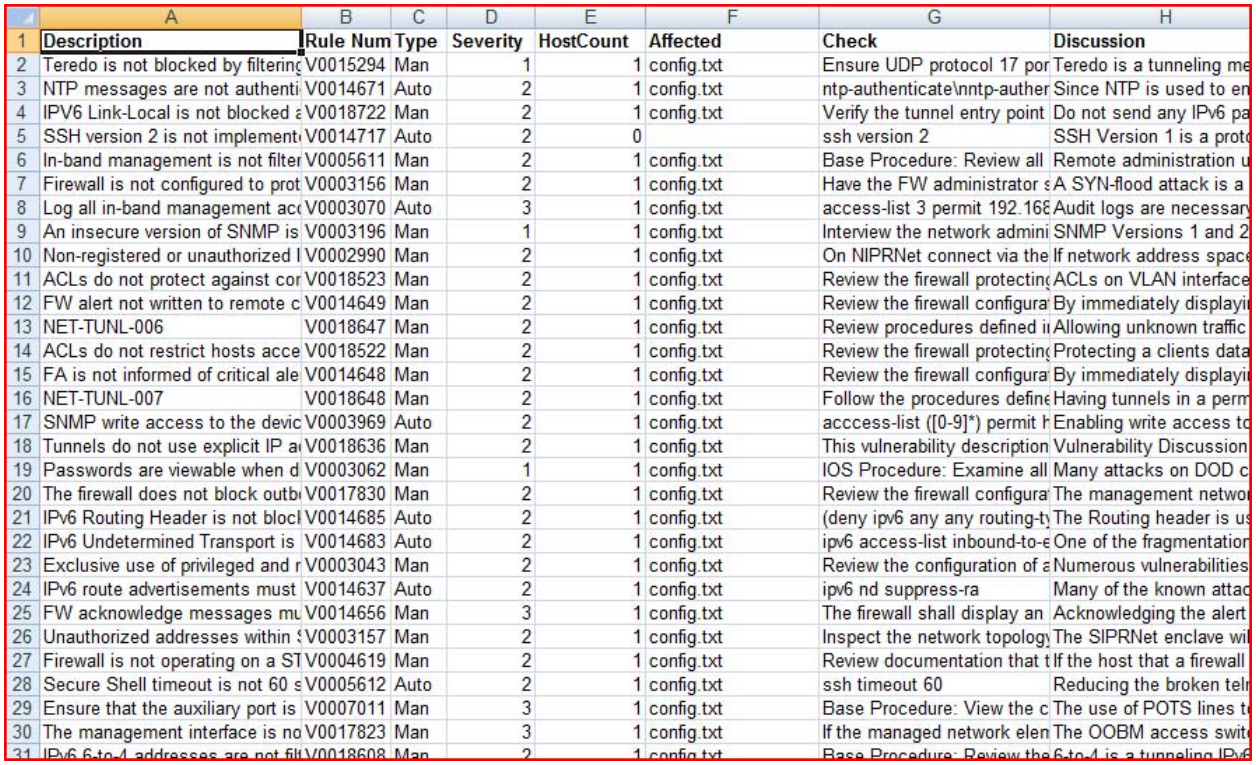
The first page in the workbook is the summary page (figure 14), which contains some basic statistics, such as the total number of flags, what level of rules were violated, and how many rules were violated at each level. We used colors and large print to make this page very visible and easily accessible.



	A	B	C	D	E
1	**Summary**				
2					
3	**Overall Findings:**				
4		**Instances**			
5	**High**	48			
6	**Medium**	284			
7	**Low**	198			
8	**Manual**	1552			
9	**Passed**	238			
10					
11	**Rules Violated by Severity:**				
12		**Instances**			
13	**High**	3			
14	**Medium**	21			
15	**Low**	16			
16	**Manual**	97			
17	**Passed**	8			
18					
19	**Number of Files at Each Severity Level:**				
20		**Instances**			
21	**High**	16			
22	**Medium**	0			
23	**Low**	0			
24	**Manual**	0			
25	**Passed**	0			
26					
27					
28					
29					

Figure 14. Summary page from generated XML report.

The second page is a findings page (figure 15), which contains a list of all the rules in the checklist and their details. The purpose of this page is to provide a place to look up any additional data the auditor may require for manual rule checking or verifying the accuracy of the program.



	A	B	C	D	E	F	G	H
1	Description	Rule Num	Type	Severity	HostCount	Affected	Check	Discussion
2	Teredo is not blocked by filterin	V0015294	Man	1	1	config.txt	Ensure UDP protocol 17 por	Teredo is a tunneling me
3	NTP messages are not authenti	V0014671	Auto	2	1	config.txt	ntp-authenticate\nntp-auther	Since NTP is used to en
4	IPV6 Link-Local is not blocked a	V0018722	Man	2	1	config.txt	Verify the tunnel entry point	Do not send any IPv6 pa
5	SSH version 2 is not implementi	V0014717	Auto	2	0		ssh version 2	SSH Version 1 is a proto
6	In-band management is not filter	V0005611	Man	2	1	config.txt	Base Procedure: Review all	Remote administration u
7	Firewall is not configured to prot	V0003156	Man	2	1	config.txt	Have the FW administrator s	A SYN-flood attack is a
8	Log all in-band management acc	V0003070	Auto	3	1	config.txt	access-list 3 permit 192.168	Audit logs are necessary
9	An insecure version of SNMP is	V0003196	Man	1	1	config.txt	Interview the network admini	SNMP Versions 1 and 2
10	Non-registered or unauthorized I	V0002990	Man	2	1	config.txt	On NIPRNet connect via the	If network address space
11	ACLs do not protect against cor	V0018523	Man	2	1	config.txt	Review the firewall protectin	ACLs on VLAN interface
12	FW alert not written to remote c	V0014649	Man	2	1	config.txt	Review the firewall configura	By immediately displayi
13	NET-TUNL-006	V0018647	Man	2	1	config.txt	Review procedures defined i	Allowing unknown traffic
14	ACLs do not restrict hosts acce	V0018522	Man	2	1	config.txt	Review the firewall protectin	Protecting a clients data
15	FA is not informed of critical ale	V0014648	Man	2	1	config.txt	Review the firewall configura	By immediately displayi
16	NET-TUNL-007	V0018648	Man	2	1	config.txt	Follow the procedures define	Having tunnels in a perm
17	SNMP write access to the devic	V0003969	Auto	2	1	config.txt	acccess-list ([0-9]*) permit h	Enabling write access to
18	Tunnels do not use explicit IP a	V0018636	Man	2	1	config.txt	This vulnerability description	Vulnerability Discussion
19	Passwords are viewable when d	V0003062	Man	1	1	config.txt	IOS Procedure: Examine all	Many attacks on DOD c
20	The firewall does not block outb	V0017830	Man	2	1	config.txt	Review the firewall configura	The management netwo
21	IPv6 Routing Header is not block	V0014685	Auto	2	1	config.txt	(deny ipv6 any any routing-ty	The Routing header is us
22	IPv6 Undetermined Transport is	V0014683	Auto	2	1	config.txt	ipv6 access-list inbound-to-e	One of the fragmentation
23	Exclusive use of privileged and r	V0003043	Man	2	1	config.txt	Review the configuration of a	Numerous vulnerabilities
24	IPv6 route advertisements must	V0014637	Auto	2	1	config.txt	ipv6 nd suppress-ra	Many of the known attac
25	FW acknowledge messages mu	V0014656	Man	3	1	config.txt	The firewall shall display an	Acknowledging the alert
26	Unauthorized addresses within S	V0003157	Man	2	1	config.txt	Inspect the network topology	The SIPRNet enclave wil
27	Firewall is not operating on a ST	V0004619	Man	2	1	config.txt	Review documentation that t	If the host that a firewall
28	Secure Shell timeout is not 60 s	V0005612	Auto	2	1	config.txt	ssh timeout 60	Reducing the broken tel
29	Ensure that the auxiliary port is	V0007011	Man	3	1	config.txt	Base Procedure: View the c	The use of POTS lines to
30	The management interface is no	V0017823	Man	3	1	config.txt	If the managed network elen	The OOBM access swit
31	IPv6 6-to-4 addresses are not fil	V0018608	Man	2	1	config.txt	Base Procedure: Review the	6-to-4 is a tunneling IPv6

Figure 15. Excerpt from the findings page of the XML report.

The third page is a breakdown of the configuration files that were scanned, and how many violations each had and how many checks must be done manually (figure 16). Our goal was to provide a way to quickly prioritize devices based on how vulnerable the program thinks they are.

	A	B	C	D	E	F
1	**Host**	**High**	**Medium**	**Low**	**Manual**	
2	3560-3-RR6.log	0	12	1	56	
3	7018-1-RR17.log	0	12	1	56	
4	3560-4-RR19.log	0	12	1	56	
5	7018-1-VDC_Productior	0	12	1	56	
6	3560-1-RR18.log	0	12	1	56	
7	3560-1-RR20.log	0	12	1	56	
8	3560-3-RR19.log	0	12	1	56	
9	3560-2-RR4.log	0	12	1	56	
10	5020-8-RR16.log	0	12	1	56	
11	3560-4-RR6.log	0	12	1	56	
12	5020-3-RR17.log	0	12	1	56	
13	3560-1-RR19.log	0	12	1	56	
14	3560-1-RR5.log	0	12	1	56	
15	3560-4-RR18.log	0	12	1	56	
16	3560-2-RR18.log	0	12	1	56	
17	7010-3-RR16.log	0	12	1	56	
18	3560-2-RR7.log	0	12	1	56	
19	6509.log	0	11	1	56	
20	3560-1-RR10.log	0	12	1	56	
21	3560-2-RR10.log	0	12	1	56	
22	3560-1-RR4.log	0	12	1	56	
23	3560-1-RR8.log	0	12	1	56	
24	3560-1-RR21.log	0	12	1	56	
25	3560-2-RR5.log	0	12	1	56	

Figure 16. Excerpt from the systems page of the XML report.

The last page is the most important one for the auditor and provides a complete breakdown of each file for each rule (figure 17). The rules are colored green or red, as determined by whether the file passed or failed the automated checks. Also under each rule are the excerpts from the file about which an auditor needs to make a decision, thus saving them from having to open each file individually. With a complete and well-written master rule set, users rarely need any other sources to do a complete analysis of system files.

	A	B	C	D	E	F	G	H
1	**File**	**Rules**	**Captures**					
2								
3	config.txt							
4		V0002990	Non-registered or unauthorized IP addresses.					
5		V0003157	Unauthorized addresses within SIPRNet enclave.					
6		V0003005	LAN addresses are not protected from the public.					
7			nat (inside) 1 access-list pnat-sslvpn					
8			nat (inside) 2 access-list pnat-aoc-syslog					
9		V0003143	Devices exist that have standard default passwords.					
10		V0003013	Login banner is non-existent or not DOD approved.					
11		V0014643	Firewall inspection is not performed adequately					
12		V0003156	Firewall is not configured to protect the network.					
13		V0003054	Firewall has unnecessary services enabled.					
14		V0003178	Firewall Admins will be logged.					
15		V0004582	Devices are not password protected for out-of-band.					
16		V0003967	Console port is not configured to timeout-10 min.					
17			console timeout 10					
18		V0007011	Ensure that the auxiliary port is disabled.					
19		V0003175	In-band management connections require passwords.					
20		V0005611	In-band management is not filtered.					
21		V0003069	Inband traffic must be secured by FIPS requirement.					
22			logging standby					
23			logging buffer-size 100000					
24			logging buffered notifications					
25			logging trap informational					
26			logging history critical					
27			logging facility 18					
28			logging host mf 140.185.217.49					
29			logging host inside 140.185.183.21					
30			no logging message 400011					
31			no logging message 400014					
32			no logging message 302015					
33			no logging message 302014					
34			no logging message 304001					
35			no logging message 302016					

Figure 17. Excerpt from the details page of the XML report.

7. Future Extensions

As mentioned previously, this program is a prototype to show proof of concept and lay the groundwork for developing a more comprehensive SRRS for network auditing. During the process of creating the NDCA, we brainstormed many ideas and were unable to implement most of them because they were beyond the scope of the project or because there was insufficient time. The following are several of the best and potentially most useful extensions.

1. **Development of additional checklists and rules**—This is the most fundamental and important extension. Developing rules takes a non-trivial amount of time, and includes transcribing, testing, and debugging. The checklists are then simpler, but also require validation when applied to a real device, which may lead to the discovery of further refinements to the rules. It has taken several weeks to generate our current rule set. That time included several revisions as well as learning how the devices work with our evolving process. In order for the master rule set to achieve maximum effectiveness, the Regex and capture statements for each rule need to be written by a SME in coordination with a Regex expert. In this way, the SME can precisely identify the lines of a configuration file that are applicable to a specific rule, and the Regex expert can craft the exact pattern necessary to extract the identified lines. As an example, we initially created the master rule set with no input from an SME. We were able to create rules that would make a pass/fail determination or extract text for an estimated 25–30% of the rules in a STIG. Working with a Cisco SME for approximately 4 h, we were able to increase that percentage to roughly 45–50% for three STIGs. Given a sufficient amount of time and expertise, it is our opinion that those percentages could rise much higher, increasing the overall efficacy of the NDCA program. New checklists can introduce upwards of 100 rules, meaning that the time requirement can vary widely based on experience and size of the task.

2. **Adding new fields in the master rule set—** The master rule set is designed with a great amount of flexibility, allowing new optional fields to be added at any time. If additional fields are added to the STIG format, or if current fields are deemed to be necessary for the report, a new optional field can be created for each rule. This new field would simply be ignored by the NDCA until such time as the program was modified to use the field in whatever manner was necessary.

3. **Machine learning—**During the automated portion of the validation process, the machine has several opportunities to make a judgment call about whether it thinks a rule is implemented correctly. Currently, this is done simply using a Regex, but we believe that there is potential for artificial intelligence to make those decisions in the future. It could also be possible to implement training by allowing the program to take part in recording results and corrections from a professional audit.

4. **Further user interaction—**A common theme while writing the rules was the desire for a few more pieces of information that only the user could provide. Allowing the user to specify information to be used in a pattern would allow greater flexibility in rule validation. The best example of this involves the Out of Band Management (OOBM) network. The IP address for the OOBM network varies for each location, whereas the rules that apply to the OOBM network in a STIG remain the same. If the IP address of the OOBM network is provided to the NDCA by the reviewing agent, these rules could be checked for compliance by the NDCA, and the applicable information could easily be provided to the

agent in the final report. We think that an additional rule field that could be used to ask the user questions would make an excellent extension.

5. **Support for multiple checklists simultaneously—**The primary reason that we have not already included support for multiple checklists in a single report is that we have not found a way to differentiate between devices by looking at their configuration files alone. Our suggestion would be to have the team that collects the configuration files name them in such a way that the NDCA can tell exactly which checklist it should use. Currently, the NDCA can only produce a report that contains information on one type of device, requiring a different report for every variety of machine on the network. With this improvement, it would be possible to generate one comprehensive report for the entire network.

6. **Multipass logic—**For the sake of simplicity and efficiency, the NDCA only performs single pass operations for checking regular expressions. Implementing support for multiple passes could greatly enrich the level of complexity that rule options can support by allowing rules to change their own behavior dynamically. For instance, it would be possible to implement conditionals and rules that make reference to results from other rules. As an example, the name of an access list could be extracted in the first pass, and then could be used in the second pass to find all statements involving that particular access list. Another example would involve determining if a specific service is activated on the device on the first pass, and, if found, using a second pass to capture information on how the device is configured. The biggest drawback, aside from the time required to implement such a system, is that it adds a great deal of overhead to the parsing and checking process. The developer would need to ameliorate some of the additional costs, or the program might become too slow for the average user.

7. **Automatic checklist generation—**The most recent checklist update was released in XML format, which makes them more accessible to programs than the previous PDF releases. This means that it would be easier to develop a secondary program, which is capable of reading in the device checklists and creating NDCA format checklist. Such a program could reduce the workload and likelihood of errors greatly.

8. **Automatic rule generation—**The idea of automatic rule generation is similar to checklist automation, but the secondary program requirements are much more complex, if not impossible. Ideally, a parser could copy plain text information fields to set up an outline for new rules, but it would almost certainly require a human to write the Regex and other supporting fields. This development would reduce the workload and help prevent duplicate rules, but could not eliminate the task completely.

9. **Further refinements to the report:** The following are suggested for refining the report:

    - Several cells in the report are grouped and meant to show information about a single device or rule. These cells could be collapsed by default or put into an expandable tree format to reduce the size of the report on the screen.

    - Whenever a capture field or other Regex puts an IP address into the report, it could be useful to also include a "whois" lookup alongside it. Auditors would then have access to knowledge, such as the country of origin, without having to go to another source.

    - Currently, all fields are static and do not reflect changes to other parts of the report. It would be a nice feature if the summary pages reflected the results of manual entries or corrections to the detailed output. For example, if the user decides that the program had a false positive, they could change it to a pass and the summary page would decrement the warnings and increment the number of passed rules.

10. **Further refinements to the GUI—**We have developed and justified the design of the GUI, but our user testing has been minimal due to time and resource constraints. The interface should be tested with a wide array of users of various technical backgrounds, and then adjusted based on feedback. The goal is to make the NDCA as accessible as possible so that it will be used to secure more network devices.

11. **Password hash checker—**Password hashes are often kept in the configuration files alongside user accounts. It would be useful to add a rudimentary password hash cracker to ensure that default or easily cracked passwords are not in use. This would be a modular extension that does not require rule sets; it simply does a single pass through the file and attempts a quick crack on any hashes it encounters. It could then raise a notification if it discovers any weaknesses.

12. **XML macros—**A further extension to the reporting process would be to add macros and other functionality that makes the data itself more accessible to other interested parties. For instance, a macro could be designed that copies all of the relevant data and auditing results into a formal report that can be distributed to non-technical personnel. Another such addition would be to allow the final auditor to certify their work by attaching a digital signature using their common access card (CAC). Most improvements like this would simply require a developer who is familiar with XML to construct the macros and then add the code to the hard coded section of the XML report generator.

8. Conclusion

The goal of this project was to create a framework for automating a time-consuming and tedious process. We feel that we have reached this goal by showing proof of concept on a subset of devices that automation is possible to a certain degree. We are planning to collaborate with other DoD components to develop this project into a Government standard, and hopefully transition this project to other agencies. The current implementation can collate data for several types of network devices, and it provides meaningful guidance for the auditors of the devices on which we have chosen to focus. Further development of existing rule sets, as well as the creation of several more by SMEs, would fully realize the usefulness of the product by improving efficiency and accuracy. We feel that the Government needs the extra security that this product can bring to our network infrastructure, and anything that decreases our vulnerability helps the Soldiers that these systems support.

List of Symbols, Abbreviations, and Acronyms

CAT	category
COTS	commercial off-the-shelf
DISA	Defense Information Systems Agency
DoD	Department of Defense
GOTS	Government off-the-shelf
GUI	graphical user interface
IASE	Information Assurance Support Environment
IDS	intrusion detection system
NDCA	Network Device Checklist Automator
OOBM	Out of Band Management
Regex	regular expressions
SA	System Administrator
SME	subject matter expert
SRRS	Security Readiness Review Script
STIG	Security Technical Implementation Guide

NO. OF COPIES	ORGANIZATION
1 ELEC	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080
1	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV & ENGRG CTR ARMAMENT ENGRG & TECHNLGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 ABERDEEN PROVING GROUND MD 21005-5001
1	PM TIMS, PROFILER (MMS-P) AN/TMQ-52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703
1	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300
1	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402
3	AARON P. HILTGEN 14017 MANDERSON PLAZA #101 OMAHA NE 68164
3	DONALD A. BENNETT 10013 S. IRVINGTON AVE TULSA OK 74137

NO. OF COPIES	ORGANIZATION
1	US ARMY RSRCH LAB ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005-5066
3	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIM L TECHL LIB ATTN RDRL CIM P TECHL PUB ADELPHI MD 20783-1197

TOTAL: 17 (1 ELEC, 1 CD, 15 HCS)

INTENTIONALLY LEFT BLANK.